BUNDESREPUBLIK DEUTSCHLAND

Prioritätsbescheinigung über die Einreichung einer Patentanmeldung

Aktenzeichen: 101 62 584.7

Anmeldetag: 19. Dezember 2001

Anmelder/Inhaber: Infineon Technologies AG, 81669 München/DE

Bezeichnung: Verfahren und Vorrichtung zum Absichern einer Exponentiations-Berechnung mittels dem chinesischen Restsatz (CRT)

Priorität: 17. Oktober 2001 DE 101 51 139.6

IPC: H 04 L 9/30

Die angehefteten Stücke sind eine richtige und genaue Wiedergabe der ursprünglichen Unterlagen dieser Patentanmeldung.

München, den 2. April 2004
Deutsches Patent- und Markenamt
Der Präsident
Im Auftrag

Stremme

A 9161
03/00
EDV-L

CERTIFIED COPY OF
PRIORITY DOCUMENT

SCHOPPE, ZIMMERMANN, STÖCKELER & ZINKLER

PATENTANWÄLTE

European Patent Attorneys
European Trademark Attorneys

Fritz Schoppe, Dipl.-Ing.
Tankred Zimmermann, Dipl.-Ing.
Ferdinand Stöckeler, Dipl.-Ing.
Franz Zinkler, Dipl.-Ing.

Telefon/Telephone 089/790445-0
Telefax/Facsimile 089/790 22 15
Telefax/Facsimile 089/74996977
e-mail: szsz_iplaw@t-online.de

Patentanwälte · Postfach 710867 · 81458 München
Infineon Technologies AG
St.-Martin-Str. 53

81669 München

Verfahren und Vorrichtung zum Absichern einer Exponentiations-Berechnung mittels dem chinesischen Restsatz (CRT)

Beschreibung

Verfahren und Vorrichtung zum Absichern einer Exponentiations-Berechnung mittels dem chinesischen Restsatz (CRT)

Die vorliegende Erfindung bezieht sich auf die Kryptographie und insbesondere auf Verfahren und Vorrichtungen zum Absichern einer Exponentiations-Berechnung mittels chinesischem Restsatz (CRT) gegen Fehlerattacken.

Die modulare Exponentiation ist eine der Kernberechnungen für verschiedene kryptographische Algorithmen. Ein Beispiel für einen weit verbreiteten kryptographischen Algorithmus ist das RSA-Kryptosystem, das beispielsweise in „Handbook of Applied Cryptography", Menezes, van Oorschot, Vanstone, CRC Press, 1996, Kapitel 8.2, beschrieben ist. Das RSA-Kryptosystem arbeitet folgendermaßen. Bei der Verschlüsselung verschlüsselt eine Partei B eine Nachricht m für eine andere Partei A. Die Partei A soll die von B erhaltene verschlüsselte Nachricht entschlüsseln. Die Partei B erhält zunächst den öffentlichen Schlüssel von der Partei A. Die Partei B stellt dann die zu verschlüsselnde Nachricht als Ganzzahl m dar. Dann verschlüsselt die Partei B die Nachricht m folgendermaßen:

$$c = m^e \bmod n \qquad (1)$$

In Gleichung (1) stellt m die Klartext-Nachricht dar. e ist der öffentliche Schlüssel. n ist der Modul und ist ebenfalls öffentlich. c stellt die verschlüsselte Nachricht dar.

Die Partei B sendet nun die verschlüsselte Nachricht c zu der Partei A.

Zur Entschlüsselung, also um den Klartext m wieder aus dem Geheimtext c zu erhalten, führt A folgende Berechnung aus:

$$m = c^d \bmod n \qquad (2)$$

In Gleichung (2) stellt d den privaten Schlüssel der Partei A dar, der vor Angriffen zu schützen ist.

In der Technik ist ferner auch ein RSA-Signaturalgorithmus bekannt. Hierbei wird folgendermaßen vorgegangen. Jede Entität A erzeugt zunächst zwei große Primzahlen p und q und berechnet dann den Modul n aus dem Produkt von p und q. Daraus wird dann, wie es ebenfalls im oben bezeichneten Fachbuch im Kapitel 11.3 beschrieben ist, eine Schlüsselerzeugung vorgenommen, so daß jede Partei einen öffentlichen Schlüssel hat, der aus n, also dem Modul, und e besteht, während jede Partei zusätzlich einen privaten Schlüssel d hat.

Zur RSA-Signaturerzeugung und Verifikation signiert die Entität A eine Nachricht m. Jede Entität B soll dann die Signatur von A verifizieren und die Nachricht m aus der Signatur wiedergewinnen.

Bei der Signaturerzeugung berechnet die Entität A zunächst eine Ganzzahl $m' = R(m)$. Danach führt die Entität A folgende Berechnung durch:

$$s = m'^{d} \bmod n \qquad (3)$$

s ist dabei die Signatur von A für die Nachricht m.

Zur Verifikation der Signatur der Partei A und zum Wiedergewinnen der Nachricht m muß die Partei B folgendermaßen vorgehen:

Zunächst muß die Partei B den öffentlichen Schlüssel (n, e) von A erhalten. Dann führt die Partei B folgende Berechnung durch:

$$m' = s^{e} \bmod n \qquad (4)$$

In Gleichung (4) ist e der öffentliche Schlüssel von A.

Die Partei B wird dann verifizieren, ob m´ das Element aus einem Raum M_R ist. Wenn dies nicht der Fall ist, wird die Signatur zurückgewiesen. Wenn dies der Fall ist, wird die Nachricht m wiedergewonnen, indem $m = R^{-1}(m´)$ berechnet wird.

Aus der obigen Darstellung wird ersichtlich, daß die modulare Exponentiation an vielerlei Stellen benötigt wird. Insbesondere wird zur RSA-Verschlüsselung in Gleichung (2) und zur RSA-Signaturerzeugung in Gleichung (3) mit dem geheimen Schlüssel d gerechnet.

Nachdem der geheime Schlüssel - genauso wie der öffentliche Schlüssel - bei typischen RSA-Systemen beträchtliche Längen annehmen kann, wie z. B. 1024 oder 2048 Bits, ist die modulare Exponentiation eine relativ aufwendige Berechnung.

Um die modulare Exponentiation schneller berechnen zu können, ist es bekannt, den sogenannten chinesischen Restsatz (CRT; CRT = Chinese Remainder Theorem) einzusetzen, der im Absatz 2.120 des oben bezeichneten Fachbuchs beschrieben ist. Für RSA-Systeme wird insbesondere der Algorithmus von Garner bevorzugt, der ebenfalls in dem oben beschriebenen Fachbuch im Abschnitt 14.5.2 beschrieben ist. Der klassische Algorithmus für den CRT benötigt typischerweise eine modulare Reduktion mit dem Modul M, während dies bei dem Algorithmus nach Garner nicht der Fall ist. Statt dessen wird hier die eine „große" modulare Exponentiation in zwei „kleine" modulare Exponentiationen aufgeteilt, deren Ergebnisse dann gemäß dem chinesischen Restsatz zusammengesetzt werden. Obwohl hier zwei Exponentiationen benötigt werden, ist es dennoch günstiger, zwei „kleine" modulare Exponentiationen zu berechnen, als eine „große" modulare Exponentiation.

Zur Darstellung des RSA-CRT-Verfahren unter Verwendung des Algorithmus von Garner wird auf Fig. 3 Bezug genommen. In ei-

nem Block 100 sind die Eingangsparameter dargelegt, die alle lediglich von p und q sowie vom Schlüssel d abhängen, jedoch nicht von der beispielsweise zu signierenden Nachricht m. In einem Block 102 ist die Ausgabe des Algorithmus dargestellt, wie sie anhand von Gleichung (2) oder Gleichung (3) dargestellt worden ist.

Aus den im Block 100 dargestellten Eingangsgrößen wird dann in einem Block 104 eine erste modulare Hilfs-Exponentiation (sp) berechnet. Analog dazu wird in einem Block 106 dann eine zweite modulare Hilfs-Exponentiation (sq) berechnet. Die Ergebnisse der ersten und der zweiten modularen Hilfs-Exponentiation werden dann in einem Block 108 gemäß dem chinesischen Restsatz zusammengesetzt, um das Ergebnis $s = m^d \bmod n$ zu erhalten. Generell ist das in Fig. 3 dargestellte RSA-CRT-Verfahren etwa um das Vier-fache schneller als die direkte Berechnung der im Block 102 dargestellten Ausgabe beispielsweise mittels des Square-and-Multiply-Algorithmus.

Aufgrund der Recheneffizienz ist der RSA-CRT-Algorithmus, der in Fig. 3 dargestellt ist, dem Square-and-Multiply-Algorithmus in jedem Fall vorzuziehen. Nachteilig am RSA-CRT-Algorithmus ist jedoch die Tatsache, daß er gegenüber kryptographischen „Angriffen" dahingehend sehr anfällig ist, daß der geheime Schlüssel d ermittelt werden kann, wenn eine fehlerhafte Berechnung des RSA-CRT-Algorithmus entsprechend ausgewertet wird. Diese Tatsache ist in „On the Importance of Eliminating Errors in Cryptographic Computations", Boneh, DeMillo, Lipton, J. Cryptology (2001) 14, S. 101 bis 119, beschrieben. Es wird ausgeführt, daß der geheime Signaturschlüssel, der bei einer Implementation des RSA-Verfahrens, das auf dem chinesischen Restsatz (CRT) basiert, aus einer einzigen fehlerhaften RSA-Signatur ermittelt werden kann.

Eine fehlerhafte RSA-Signatur kann dadurch erhalten werden, daß die Software oder die Hardware, die den Algorithmus ausführt, zu Fehlern gebracht wird, beispielsweise durch Ausset-

zen des Kryptoprozessors gegenüber einer mechanischen oder thermischen Belastung.

Als Gegenmaßnahmen gegen solche Angriffe, die auf Hardware-Fehlern basieren, wird vorgeschlagen, die Ausgabe jeder Berechnung zu überprüfen, bevor dieselbe aus dem Chip ausgegeben wird. Obwohl dieser zusätzliche Verifikationsschritt das Systemverhalten verschlechtern kann, wird davon gesprochen, daß diese zusätzliche Verifikation aus Sicherheitsgründen wesentlich ist.

Die einfachste Art und Weise der Verifikation besteht darin, eine Gegenrechnung mit dem öffentlichen Exponenten e durchzuführen, wobei folgende Identität festgestellt werden soll:

$$(m^d)^e = m \bmod n \qquad (5)$$

Dieser zusätzliche Verifikationsschritt ist jedoch vom Rechenaufwand her unmittelbar vergleichbar mit dem eigentlichen Signatur- bzw. Entschlüsselungs-Schritt und führt daher zu einer starken Verringerung des Systemverhaltens, liefert jedoch eine hohe Sicherheit.

Nachteilig ist jedoch, daß der öffentliche Schlüssel e in üblichen Protokollen, wie z. B. der ZKA-lib, nicht verfügbar ist. Die ZKA-lib ist eine Sammlung von Spezifikationen des zentralen Kreditausschusses, die regeln, welche Daten verfügbar sind. Für das RSA-CRT-Verfahren sind lediglich die im Block 100 von Fig. 3 gegebenen Eingangsdaten verfügbar. Der öffentliche Schlüssel e ist hierbei nicht Teil der in der ZKA-lib-Beschreibung vorgegebenen Parameter. Der Exponent e müßte daher aufwendig berechnet werden, um die „Gegenrechnung" gemäß Gleichung (5) durchführen zu können. Dies würde die Leistung der Signatur-Chipkarte weiter reduzieren und dürfte dazu führen, daß solche Algorithmen aufgrund ihrer langsamen Arbeitsweise keine Chance auf eine Durchsetzung am Markt haben.

In der Fachveröffentlichung von A. Shamir, „How to check modular Exponentiation", Rump Session, Eurocrypt 97, ist ein weiteres Verfahren beschrieben, um Signaturen zu verifizieren, die durch RSA-CRT-Verfahren erzeugt werden. In dieser Fachveröffentlichung wird vorgeschlagen, eine kleine Zufallszahl r (beispielsweise 32 Bits) zu verwenden und statt der Berechnung im Block 104 folgende Berechnung auszuführen:

$$sp' = m^d \bmod pr \tag{6}$$

Statt dem Block 106 wird folgende Berechnung ausgeführt:

$$sp' = m^d \bmod qr \tag{7}$$

Dann, unmittelbar nach den Berechnungen gemäß den Gleichungen (6) und (7) werden folgende Überprüfungsberechnungen durchgeführt:

$$sp' \bmod r = sq' \bmod r \tag{8}$$

Wenn die Überprüfung gemäß Gleichung (8) wahr ist, wird sp und sq aus folgender Gleichung (9) erhalten:

$$sp' \bmod p = sp \quad ; \quad sq' \bmod q = sq \tag{9}$$

Aus den durch Gleichung (9) erhaltenen Werten sp und sq wird dann die im Block 108 in Fig. 3 dargestellte Berechnung durchgeführt, um aus den modularen Hilfs-Exponentiationen das Gesamtergebnis s mittels des chinesischen Restsatzes zusammenzufügen.

Nachteilig an diesem Verfahren ist, daß zur Überprüfung lediglich der Hilfsparameter r sowie die Zwischenergebnisse sp' und sq' herangezogen werden, wobei die Überprüfung nicht zur Unterdrückung eines Ausgabewerts führt, wenn eine kryptographische Attacke stattgefunden hat, die möglicherweise

nicht die Zwischenergebnisse sp´, sq´ oder den Parameter r beeinträchtigt hat, aber dann, beispielsweise in den in Gleichung (9) gegebenen Schritten und der abschließenden Zusammensetzung des Algorithmus zu einem Hardware-Fehler führt, der dazu verwendet werden kann, um den geheimen Schlüssel d unerlaubterweise auszuspähen.

Darüber hinaus wird in der zitierten Fachveröffentlichung von Boneh u. a. beispielsweise als Abwehrmaßnahme zur Sicherung des Fiat-Shamir-Schemas vorgeschlagen, Registerfehler, die auftreten, während der Prozessor auf eine Antwort von außen wartet, dadurch abzuwehren, daß Fehlererfassungsbits zum Schutz des internen Speichers eines Prozessors eingesetzt werden. Weitere Maßnahmen, um RSA-Signaturen zu schützen, bestehen darin, eine Zufälligkeit in das Signaturverfahren einzuführen. Die Zufälligkeit stellt sicher, daß der Unterzeichner niemals die gleiche Nachricht zweimal unterzeichnet. Ferner weiß der Verifizierer, wenn er eine fehlerhafte Signatur vorliegen hat, nicht den vollständigen Klartext, der unterzeichnet worden ist.

Die Aufgabe der vorliegenden Erfindung besteht darin, ein verbessertes Konzept zum Absichern einer Exponentiations-Berechnung mittels chinesischem Restsatz (CRT) gegen Fehlerattacken zu schaffen.

Diese Aufgabe wird durch ein Verfahren gemäß Anspruch 1 oder eine Vorrichtung gemäß Anspruch 11 gelöst.

Der vorliegenden Erfindung liegt die Erkenntnis zugrunde, daß ein Sicherheitsleck dahingehend besteht, wenn das Zusammenfügen zweier Hilfsgrößen, die modulare Exponentiationen mit einem „kleinen" Modul sind, mittels chinesischem Restsatz, um das Ergebnis einer modularen Exponentiation mit einem „großen" Modul zu erhalten, nicht überprüft wird. Wenn lediglich die Berechnung der Hilfsgrößen überprüft wird, jedoch dann, im Zusammenfügungsschritt, keine Überprüfung mehr vorgenommen

wird, kann eine Fehlerattacke, die erst nach der Berechnung der Hilfsgrößen zu einer Fehlfunktion des Rechenwerks im Kryptographieprozessor führt, zu einer falschen Ausgabe führen. Insbesondere RSA-Berechnungen mittels dem chinesischen Restsatz sind aus Effizienzgründen erwünscht, da sie einen Rechenzeitgewinn um einen Faktor 4 ermöglichen. Andererseits sind RSA Berechnungen mit CRT besonders anfällig für Sicherheitslecks. Schließlich soll die Überprüfung des Zusammenfügungsschritts nicht besonders aufwendig sein, um nicht den Rechenzeitgewinn durch Verwendung des CRT durch das erneute Berechnen des Zusammenfügungsschritts wieder zunichte gemacht werden. Erfindungsgemäß wird daher nach dem Zusammenfügen der ersten Hilfsgröße und der zweiten Hilfsgröße, um ein Ergebnis der Exponentiations-Berechnung zu erhalten, das Ergebnis der Exponentiations-Berechnung mittels eines Prüfalgorithmus überprüft, der sich von dem Zusammenfügungsalgorithmus unterscheidet, und der auf eine erste Primzahl bzw. eine zweite Primzahl zugreift. Falls die Überprüfung ergibt, daß der Prüfalgorithmus ein anderes als ein vorbestimmtes Ergebnis liefert, wird die Ausgabe des Ergebnisses der Exponentiations-Berechnung unterdrückt. Ansonsten kann davon ausgegangen werden, daß keine Hardware-Attacke stattgefunden hat, so daß das Ergebnis der Exponentiations-Berechnung ausgegeben werden kann.

Ein Vorteil der vorliegenden Erfindung besteht darin, daß das Sicherheitsleck, das bisher beim Zusammenfügen mittels chinesischem Restsatz bestand, „gestopft" wird.

Ein weiterer Vorteil der vorliegenden Erfindung besteht darin, daß zur Überprüfung keine erneute Zusammenfügungsberechnung stattfinden muß, sondern daß mit einfachen Mitteln unter Verwendung der ersten bzw. der zweiten Primzahl eine Überprüfung der Exponentiations-Berechnung durchgeführt werden kann.

Ein weiterer Vorteil der vorliegenden Erfindung besteht darin, daß nunmehr nicht irgendwo in der Mitte des RSA-CRT-

Algorithmus, sondern unmittelbar vor der Ausgabe des für einen Angreifer relevanten Ergebnisses eine Sicherheitsüberprüfung durchgeführt wird.

Ein weiterer Vorteil der vorliegenden Erfindung besteht darin, daß die Überprüfung des Zusammenfügungsschritts gemäß der vorliegenden Erfindung ohne weiteres mit weiteren Überprüfungsmaßnahmen, beispielsweise zur Überprüfung der Ergebnisse der Hilfs-Exponentiationen oder aber zur Abwehrung von Hardware-Attacken durch Überprüfung der Eingangsdaten nach der kryptographischen Berechnung kombiniert werden können, um einen nach allen Seiten abgesicherten RSA-CRT-Algorithmus zu erhalten, bei dem der Mehraufwand zur Absicherung des Algorithmus gegenüber Hardware-Attacken im Vergleich zum Gewinn durch Verwendung des RSA-CRT-Verfahrens klein ist.

Bevorzugte Ausführungsbeispiele der vorliegenden Erfindung werden nachfolgend Bezug nehmend auf die beiliegenden Zeichnungen detailliert erläutert. Es zeigen:

Fig. 1 ein Blockschaltbild des erfindungsgemäßen Konzepts;

Fig. 2 ein bevorzugtes Ausführungsbeispiel für ein sicheres RSA-CRT-Verfahren, bei dem neben der Überprüfung des Zusammenfügungsschritts auch eine Überprüfung einer Veränderung von Eingangsdaten und eine Überprüfung von Zwischenergebnissen vorgenommen wird; und

Fig. 3 ein Blockschaltbild des RSA-CRT-Verfahrens unter Verwendung des Algorithmus von Garner.

Fig. 1 zeigt ein Blockdiagramm des erfindungsgemäßen Konzepts zum Absichern einer Exponentiations-Berechnung mittels chinesischem Restsatz. Zunächst werden zwei Primzahlen p, q bereitgestellt, deren Produkt gleich dem Modul n ist (Block 10). Hierauf wird unter Verwendung der ersten Primzahl p und

unter Verwendung des chinesischen Restsatzes eine erste Hilfsgröße sp berechnet (Block 12). Analog wird unter Verwendung der zweiten Primzahl q ebenfalls mittels des CRT eine zweite Hilfsgröße sq berechnet. Die erste und die zweite Hilfsgröße sp, sq werden nunmehr zusammengefügt (Block 16), um aus den beiden Hilfs-Exponentiationen sp und sq mit jeweils „kleinem" Modul p bzw. q eine Exponentiation mit „großem" Modul n zu erhalten. Der beste Gewinn wird dann erreicht, wenn die beiden Primzahlen p, q etwa die gleiche Länge haben, also jeweils halb so groß sind wie der „große" Modul n.

Erfindungsgemäß wird nunmehr in einem Block 18 die Zusammenfügung der ersten und der zweiten Hilfsgröße mittels eines Überprüfungsalgorithmus überprüft, der sich von dem Zusammenfügungsalgorithmus, der im Block 16 ausgeführt wird, unterscheidet, und der das Ergebnis s der Zusammenfügung und die erste Primzahl und/oder die zweite Primzahl verwendet, wie es durch Primzahl-Eingangsleitungen 20a, 20b symbolisch dargestellt ist. Ergibt die Überprüfung im Block 18, daß der in Block 18 verwendete Prüfalgorithmus ein anderes als ein vorbestimmtes Ergebnis liefert, wird zu einem Block 22 gesprungen, der eine Ausgabe des Ergebnisses des Zusammenfügungsschritts 16 unterdrückt. Ergibt der Überprüfungsalgorithmus im Block 18 dagegen, daß sich das vorbestimmte Resultat ergibt, so kann zu einem Block 24 gesprungen werden, um eine Ausgabe des Ergebnisses des Zusammenfügungsschritts 16 zu veranlassen. Diese Ausgabe kann beispielsweise eine digitale Signatur oder ein entschlüsselter Klartext sein.

Bei einem bevorzugten Ausführungsbeispiel der vorliegenden Erfindung lautet der Überprüfungsalgorithmus folgendermaßen:

$$s \bmod p = sp \text{ und/oder} \tag{10}$$

$$s \bmod q = sq \tag{11}$$

Dieser Überprüfungsalgorithmus kann unmittelbar auf den Zusammenfügungsschritt im Block 108 von Fig. 3 angewendet werden. Hierzu wird folgendermaßen vorgegangen. Zunächst wird das Zwischenergebnis s, das der Block 108 liefert, und das beispielsweise in einem Ausgangsregister des Kryptoprozessors vorliegt, verwendet. Darüber hinaus wird an eine Eingangsdaten-Speicherstelle zugegriffen, an der die erste Primzahl p abgespeichert ist. Hierauf wird s mod p berechnet und das erhaltene Ergebnis zwischengespeichert. Dann wird auf eine Zwischenergebnis-Speicherstelle zugegriffen, an der das Ergebnis des Blocks 104, als sp, gespeichert ist. Das Ergebnis der Berechnung s mod p wird mit dem gespeicherten sp verglichen. Stellt sich heraus, daß die Gleichheitsbedingung erfüllt ist, so hat der erste Teil des Überprüfungsalgorithmus zu keinem Fehler geführt. Die analoge Vorgehensweise kann mit der zweiten Primzahl q durchgeführt werden, um s mod q zu berechnen, und um dieses Ergebnis dann mit sq zu vergleichen. Ergibt sich auch hier die Gleichheitsbedingung, so kann davon ausgegangen werden, daß der Zusammenfügungsschritt korrekt stattgefunden hat, und daß auch an der Eingangsdaten-Speicherstelle, wo p und q gespeichert sind, als auch an der Zwischenergebnisspeicherstelle, wo sp und sq gespeichert sind, keine Veränderung aufgetreten ist, die auf eine Hardware-Attacke hindeuten würde. Ergibt dagegen zumindest ein Teil des Überprüfungsalgorithmus einen Fehler, so wird die Ausgabe unterdrückt, da bei der Zusammenfügung selbst und/oder an den Speicherplätzen für sp, sq, p, q, Veränderungen aufgetreten sind.

Im nachfolgenden wird der erste Teil des Überprüfungsalgorithmus s mod p = sp näher erläutert. Aus der im Block 108 dargestellten Gleichung wird ersichtlich, daß dieselbe nach einer modularen Reduktion mit der ersten Primzahl p zu s = sq + (sp - sq) „entartet", so daß sich die Terme +sq und -sq gegenseitig aufheben, und daß sp verbleibt.

Wird dagegen mit dem zweiten Teil des Überprüfungsalgorithmus mit s mod q = sq gearbeitet, so ergibt sich bei näherer Betrachtung der im Block 108 befindlichen Gleichung, daß diese den Wert sq + einem ganzzahligen Faktor multipliziert mit q ist, wobei die modulare Reduktion mittels des Moduls q von q mal einer konstanten Zahl den Wert 0 ergibt, so daß aus der im Block 108 befindlichen Gleichung noch sq verbleibt.

Aus näherer Analyse der im Block 108 befindlichen Gleichung für s können unter Verwendung der obigen Ausführungen weitere Überprüfungsalgorithmen abgeleitet werden, um das Ergebnis der Zusammenfügung auf irgendeine Art und Weise zu verarbeiten, um ein vorbestimmtes Resultat zu erhalten, wenn kein Fehler aufgetreten ist, bzw. um ein von dem vorbestimmten Resultat abweichendes Resultat zu erhalten, wenn ein Fehler beim Zusammenfügen aufgetreten ist.

Im nachfolgenden wird anhand von Fig. 2 ein bevorzugtes Ausführungsbeispiel der vorliegenden Erfindung dargestellt, bei dem der Zusammenfügungsschritt als Block 64 gekennzeichnet ist, während der erste Teil des Überprüfungsalgorithmus im Block 66a dargestellt ist, und der zweite Teil des Überprüfungsalgorithmus im Block 66b dargestellt ist. Das in Fig. 2 gezeigte gesamte RSA-CRT-Verfahren verwendet ferner zur weiteren Absicherung gegenüber Fehlerattacken eine Fehlerüberprüfung anhand der Eingangsdaten als auch anhand der ersten und der zweiten Hilfsgröße, um die korrekte Berechnung dieser Größen nachzukontrollieren.

Ferner wird zusätzlich bei dem in Fig. 2 gezeigten bevorzugten Ausführungsbeispiel zum sicheren Ausführen des RSA-CRT-Verfahrens eine Überprüfung der Eingangsdaten vor der Ausgabe von Ausgangsdaten eines kryptographischen Algorithmus an mehreren Stellen innerhalb des Algorithmus verwendet.

Des weiteren wird bei dem in Fig. 2 gezeigten Ausführungsbeispiel auch die Berechnung des kryptographischen Algorithmus

selbst, und zwar insbesondere die Berechnung der beiden Hilfs-Exponentiationen überprüft.

Zunächst werden, wie es bereits anhand von Fig. 3 dargestellt worden ist, die Parameter p, q, dp, dq, qinv bereitgestellt, die die üblichen Eingabeparameter für das RSA-CRT-Verfahren sind. Ferner werden, wie es in einem Block 50 von Fig. 2 dargestellt ist, die zu verschlüsselnde Nachricht m sowie eine Zahl t und eine Zufallszahl rand als Eingangsdaten bereitgestellt. Vorzugsweise ist die Zahl t eine Primzahl, und vorzugsweise eine kleine Primzahl, welche beispielsweise nicht länger als 16 Bits ist, um den Vorteil des CRT-Verfahrens nicht zu stark zu schmälern, nämlich daß die beiden Hilfs-Exponentiationen mit kleinerem Modul im Vergleich zu einer einzigen modularen Exponentiation mit dem Modul n = p mal q stattfinden. Ist die Zahl t keine Primzahl, so ist dieser Fall ebenfalls möglich, in den Gleichungen müßte jedoch dann der Ausdruch (t-1) durch die Eulersche Phi-Funktion von t ersetzt werden.

Zunächst werden Eingangsdaten in Blöcken 52a, 52b verarbeitet. Als Verarbeitungsalgorithmus wird die Multiplikation des ursprünglichen Parameters p bzw. q mit der Primzahl t verwendet. Ferner wird als Verarbeitungsvorschrift die Addition von dp mit dem Produkt aus der Zufallszahl rand und der Zahl (p-1) bzw. entsprechend für q verwendet.

Es sei darauf hingewiesen, daß prinzipiell auch eine einzige der vier in den Blöcken 52a, 52b gegebenen Verarbeitungsvorschriften einen erfindungsgemäßen Effekt ergeben würde. Nach der Vollendung der Blöcke 52a, 52b werden die durch die Verarbeitung erhaltenen Sicherheitsinformationen p´, dp´, q´ und dq´ an einer Sicherheitsinformationen-Speicherstelle gespeichert. Diese Speicherstelle könnte beispielsweise der Arbeitsspeicher eines Kryptoprozessors sein, oder ein inneres Register, das dem Rechenwerk des Kryptoprozessors zugeordnet ist. Dann wird durch das Rechenwerk, wie es durch Blöcke 54a,

54b dargestellt ist, als Berechnung innerhalb des kryptographischen Algorithmus sowohl die erste Hilfs-Exponentiation (sp´) als auch die zweite Hilfs-Exponentiation (sq´) durchgeführt, wie es in Fig. 2 gezeigt ist. Nach dem Durchführen der Blöcke 54a, 54b werden die Ausgangsdaten der Berechnungen, nämlich sp´ und sq´ nicht unmittelbar entweder ausgegeben bzw. für eine weitere Berechnung weitergegeben, sondern es wird erfindungsgemäß zunächst in Blöcken 56a, 56b mittels eines Kontrollalgorithmus überprüft, ob die Eingangsdaten für die Berechnung in den Blöcken 54a, 54b während der Berechnung durch die Blöcke 54a, 54b verändert worden sind. Hierzu wird als Kontrollalgorithmus eine modulare Reduktion verwendet, wobei als vorbestimmtes Ergebnis entweder 0 erwartet wird, wie es in den ersten Zeilen der beiden Blöcke 56a, 56b dargestellt ist, oder entweder dp oder dq als vorbestimmtes Resultat erwartet wird. Das vorbestimmte Resultat ergibt sich, wenn die Größe p´, die in der Terminologie der vorliegenden Erfindung die Sicherheitsinformation ist, nicht beispielsweise durch eine Fehlerattacke verändert worden ist. Dasselbe gilt für die weitere Sicherheitsinformation dp´.

Sind die Überprüfungen in den Blöcken 56a, 56b erfolgreich, also werden vorbestimmte Ergebnisse durch den Kontrollalgorithmus erhalten, so wird zu Blöcken 58a, 58b weitergegangen. Die Blöcke 58a, 58b zeigen bevorzugte Vorberechnungen, um neben dem Eingangsdaten-Überprüfungskonzept auch ein Ergebnisdaten-Überprüfungskonzept durchzuführen. Mittels eines Ergebnis-Kontrollalgorithmus (Block 60 in Fig. 2) wird dann überprüft, ob die Berechnung der Hilfs-Exponentiationen in den Blöcken 54a, 54b korrekt stattgefunden hat.

In Blöcken 62a, 62b werden die Hilfs-Exponentiationen der Blöcke 54a, 54b entsprechend modular reduziert, um den Einfluß des Parameters t bzw. der Zufallszahl zu eliminieren. In einem Block 64 wird schließlich, wie es anhand des Blocks 108 von Fig. 3 klargestellt worden ist, der Zusammensetzungs-

schritt ausgeführt, um aus den Hilfs-Exponentiationsergebnisse sp, sq die signierte Nachricht s zu erzeugen.

Bei einem bevorzugten Ausführungsbeispiel der vorliegenden Erfindung wird dieses Ergebnis jedoch nicht unmittelbar verwendet, sondern es wird nach dem Zusammensetzen durch den Block 64 eine Überprüfung dahingehend durchgeführt, ob das Zusammensetzen erfolgreich war.

Dies wird dadurch erreicht, daß zunächst die erhaltene signierte Nachricht s unter Verwendung der Primzahl p als Modul modular reduziert wird. Dieser Kontrollalgorithmus sollte als Ergebnis sp ergeben, wobei dieses sp gleich dem im Block 62a ausgerechneten Wert sp sein muß.

Analog wird in einem Block 66b vorgegangen, um die Korrektheit des Ergebnisses s auch anhand einer modularen Reduktion mit der Primzahl q als Modul zu überprüfen. Hierzu wird zur Ausführung der in Block 66a gegebenen Berechnung zunächst auf die Zwischenspeicherstelle zugegriffen, an der das Ergebnis des Blocks 64 abgespeichert wurde. Zusätzlich wird auf die Speicherstelle zugegriffen, an der das Eingangsdatum p gespeichert ist. Schließlich wird, um den Vergleich des Blocks 66a durchzuführen, auf die Speicherstelle zugegriffen, in der das Ergebnis des Blocks 62a, also sp, gespeichert ist. Analog wird im Block 66b für s, q und sq vorgegangen.

Liefert die Berechnung im Block 66a ein vorbestimmtes Resultat dahingehend, daß die linke und die rechte Seite der im Block 66a gegebenen Gleichung nicht gleich sind, so wird ein Fehler ausgegeben, und die Ausgabe des Ergebnisses s des Blocks 64 wird unterdrückt. Dieselbe Unterdrückung des Ergebnisses s findet statt, wenn die Berechnung im Block 66b ergibt, daß ein Fehler stattgefunden hat. Eine Unterdrückung findet somit vorzugsweise bereits dann statt, wenn ein einziger Block einen Fehler ergeben hat bzw., in anderen Worten ausgedrückt, findet eine Ergebnisausgabe mittels eines Blocks

68 nur dann statt, wenn sowohl die Berechnung im Block 66a als auch die Berechnung im Block 66b korrekt waren.

Anhand des Beispiels in Block 66a wird deutlich, daß dieser Ergebnis-Kontrollalgorithmus dahingehend vorteilhaft ist, daß er unmittelbar das Ergebnis des Blocks 64 zur Überprüfung verwendet, daß er jedoch auch auf den Eingangsdaten-Speicherbereich zugreift, um die Primzahl p zu erhalten bzw. den Inhalt der Speicherstelle, an der p stehen sollte, und daß zusätzlich auch ein Zwischenergebnis verwendet wird, nämlich sp, das im Schritt 62a erhalten worden ist. Mittels einer Berechnung wird somit sowohl überprüft, ob sich Eingangsdaten verändert haben, als auch wird überprüft, ob der Zusammensetzungsschritt 64 des RSA-CRT-Verfahrens von dem Krypto-Rechenwerk korrekt durchgeführt worden ist. Schließlich wird auch ein Zwischenergebnis sp verwendet, so daß in eine einzige einfache Berechnung auch Zwischenergebnis-Register mit einbezogen werden.

Aus dem in Fig. 2 gezeigten Ausführungsbeispiel wird deutlich, daß sowohl der Verarbeitungsalgorithmus, um die Sicherheitsinformationen zu erzeugen, als auch der Kontrollalgorithmus zum Überprüfen der Eingangsdaten einfache Algorithmen sind, die ohnehin in einem Krypto-Rechenwerk vorhanden sind, wie z. B. ein Multiplikationsalgorithmus oder ein Algorithmus zur Durchführung einer modularen Reduktion. Dasselbe trifft zu für die Verarbeitungsalgorithmen in den Blöcken 62a, 62b, die ebenfalls auf einer modularen Reduktion basieren, und auch für den Kontrollalgorithmus in den Blöcken 66a, 66b, der wiederum auf einer modularen Reduktion basiert.

Obgleich in dem vorhergehenden in Fig. 2 gezeigten Ausführungsbeispiel als Verarbeitungsalgorithmus die Multiplikation einer Zahl mit einer Konstanten, und als - dazu korrespondierender - Kontrollalgorithmus die modulare Reduktion des Multiplikationsergebnisses mit der ursprünglichen Zahl dargestellt worden sind, ist es für Fachleute offensichtlich, daß

eine Vielzahl von miteinander korrespondierenden Verarbeitungsalgorithmen und Kontrollalgorithmen existiert, die es ermöglichen, zu überprüfen, ob Eingangsdaten während der Durchführung einer Berechnung in einem kryptographischen Algorithmus z. B. durch Fehlerattacken verändert worden sind.

Aus Fig. 2 wird ferner deutlich, daß die Verarbeitungsalgorithmen genauso wie die Kontrollalgorithmen sehr einfach gestaltet werden können, und keine zusätzlichen Parameter benötigen, als die ohnehin vorhandenen Parameter. Insbesondere wird es erfindungsgemäß bevorzugt, nicht zusätzliche Parameter, wie z. B. den öffentlichen Schlüssel e, zunächst aufwendig zu berechnen und dann für eine „Gegenrechnung" zu verwenden, sondern möglichst viele Eingangsdaten, Zwischenergebnisdaten etc. miteinander zu verknüpfen, da damit mittels eines einzigen Überprüfungsschritts mögliche Fehler im Arbeitsspeicher, in den inneren Registern oder in dem Rechenwerk selbst detektiert werden können, um im Falle eines Fehlers eine Datenausgabe zu unterdrücken, damit keine geheimen Informationen aus einer falschen Ausgabe ermittelbar sind.

Patentansprüche

1. Verfahren zum Absichern einer Exponentiations-Berechnung mittels chinesischem Restsatz (CRT) unter Verwendung zweier Primzahlen (p, q), die Hilfs-Module für eine Berechnung von Hilfsgrößen bilden, die zusammensetzbar sind, um eine modulare Exponentiation für einen Modul zu berechnen, der gleich dem Produkt der Hilfsgrößen ist, mit folgenden Schritten:

Berechnen (12) der ersten Hilfsgröße (sp) unter Verwendung der ersten Primzahl (p) als Modul;

Berechnen (14) der zweiten Hilfsgröße (sq) unter Verwendung der zweiten Primzahl (q) als Modul;

Zusammenfügen (16) der ersten Hilfsgröße (sp) und der zweiten Hilfsgröße (sq) unter Verwendung eines Zusammenfügungsalgorithmus, um ein Ergebnis der Exponentiations-Berechnung zu erhalten;

nach dem Schritt des Zusammenfügens, Überprüfen (18) des Ergebnisses der Exponentiations-Berechnung mittels eines Prüfalgorithmus (66a, 66b), der sich von dem Zusammenfügungsalgorithmus unterscheidet, unter Verwendung der ersten Primzahl (p) und/oder der zweiten Primzahl (q), wobei der Prüfalgorithmus ein vorbestimmtes Resultat liefert, wenn der Schritt des Zusammenfügens (16) korrekt abgelaufen ist; und

falls der Schritt des Überprüfens (18) ergibt, daß der Prüfalgorithmus ein anderes als das vorbestimmte Resultat liefert, Unterdrücken (22) einer Ausgabe des Ergebnisses der Exponentiations-Berechnung.

2. Verfahren nach Anspruch 1, bei dem der Prüfalgorithmus als Eingangsdaten neben dem Ergebnis (s) der Exponentiations-Berechnung einen Inhalt einer Speicherstelle verwendet, an der die erste Hilfsgröße (sp), die zweite Hilfsgröße (sq),

die erste Primzahl (p) oder die zweite Primzahl (q) gespeichert sind.

3. Verfahren nach Anspruch 1 oder 2,

bei dem die Exponentiations-Berechnung eine RSA-Verschlüsselung, eine RSA-Entschlüsselung, eine RSA-Signaturberechnung oder eine RSA-Signatur-Verfikationsberechnung ist.

4. Verfahren nach einem der vorhergehenden Ansprüche,

bei dem der Zusammenfügungsalgorithmus der Algorithmus nach Garner ist.

5. Verfahren nach einem der vorhergehenden Ansprüche,

bei dem der Prüfalgorithmus eine modulare Reduktion des Ergebnisses der Exponentiations-Berechnung mit der ersten Primzahl (p) und/oder der zweiten Primzahl (q) als Modul umfaßt.

6. Verfahren nach einem der vorhergehenden Ansprüche,

bei dem die erste Hilfsgröße folgendermaßen berechnet wird:

$$sp := m^{dp} \bmod p;$$

bei dem die zweite Hilfsgröße folgendermaßen berechnet wird:

$$sq = m^{dq} \bmod q;$$

bei dem der Zusammenfügungsalgorithmus folgendermaßen definiert ist:

$$s = sq + \{[(sp - sq) \cdot qinv] \bmod p\} \cdot q \; ; \text{ und}$$

bei dem der Überprüfungsalgorithmus folgendermaßen definiert ist:

s mod p = sp; und/oder

s mod q = sq; und

bei dem das vorbestimmte Resultat eine Gleichheitsbedingung in dem Überprüfungsalgorithmus ist.

7. Verfahren nach einem der Ansprüche 1 bis 5, das ferner folgenden Schritt aufweist:

nach dem Schritt des Zusammenfügens der ersten Hilfsgröße und der zweiten Hilfsgröße, Überprüfen, ob Eingangsdaten für die Exponentiations-Berechnung verändert wurden, und, wenn dies der Fall ist, Unterdrücken des Ergebnisses der Exponentiations-Berechnung.

8. Verfahren nach Anspruch 7, bei dem zur Überprüfung von Hilfs-Exponenten (dp, dq) eine Zufallszahl (rand) verwendet wird.

9. Verfahren nach Anspruch 7 oder 8, bei dem zur Überprüfung der ersten Primzahl (p) und der zweiten Primzahl (q) als Eingangsdaten eine Primzahl (t) verwendet wird.

10. Verfahren nach Anspruch 9, bei der die Primzahl (t) eine Stellenzahl hat, die kleiner als eine Stellenzahl der ersten Primzahl (p) und der zweiten Primzahl (q) ist.

11. Vorrichtung zum Absichern einer Exponentiations-Berechnung mittels chinesischem Restsatz (CRT) unter Verwendung zweier Primzahlen (p, q), die Hilfs-Module für eine Berechnung von Hilfsgrößen bilden, die zusammensetzbar sind, um eine modulare Exponentiation für einen Modul zu berechnen,

der gleich dem Produkt der Hilfsgrößen ist, mit folgenden Merkmalen:

einer Einrichtung zum Berechnen (12) der ersten Hilfsgröße (sp) unter Verwendung der ersten Primzahl (p) als Modul;

einer Einrichtung zum Berechnen (14) der zweiten Hilfsgröße (sq) unter Verwendung der zweiten Primzahl (q) als Modul;

einer Einrichtung zum Zusammenfügen (16) der ersten Hilfsgröße (sp) und der zweiten Hilfsgröße (sq) unter Verwendung eines Zusammenfügungsalgorithmus, um ein Ergebnis der Exponentiations-Berechnung zu erhalten;

einer Einrichtung zum Überprüfen (18) des Ergebnisses der Exponentiations-Berechnung mittels eines Prüfalgorithmus (66a, 66b), der sich von dem Zusammenfügungsalgorithmus unterscheidet, unter Verwendung der ersten Primzahl (p) und/oder der zweiten Primzahl (q), wobei der Prüfalgorithmus ein vorbestimmtes Resultat liefert, wenn die Einrichtung zum Zusammenfügen (16) ein korrektes Ergebnis geliefert hat; und

einer Einrichtung zum Unterdrücken (22) einer Ausgabe des Ergebnisses der Exponentiations-Berechnung, falls die Einrichtung zum Überprüfen (18) anzeigt, daß der Prüfalgorithmus ein anderes als das vorbestimmte Resultat liefert.

Zusammenfassung

Verfahren und Vorrichtung zum Absichern einer Exponentiations-Berechnung mittels dem chinesischen Restsatz (CRT)

Bei einem Verfahren zum Absichern einer Exponentiations-Berechnung mittels chinesischem Restsatz wird insbesondere der Zusammenfügungsschritt (16), bei dem vorzugsweise der Zusammenfügungsalgorithmus nach Garner verwendet wird, auf seine Korrektheit hin vor der Ausgabe (24) des Ergebnisses des Zusammenfügungs-Schritts überprüft (18). Damit wird unmittelbar vor der Ausgabe des Ergebnisses der Exponentiations-Berechnung der Zusammenfügungs-Algorithmus überprüft, um die Ausgaben eines falschen Ergebnisses beispielsweise aufgrund einer Hardware-Fehlerattacke zu unterbinden, um die Fehlerattacke abzuwehren.

Figur 1

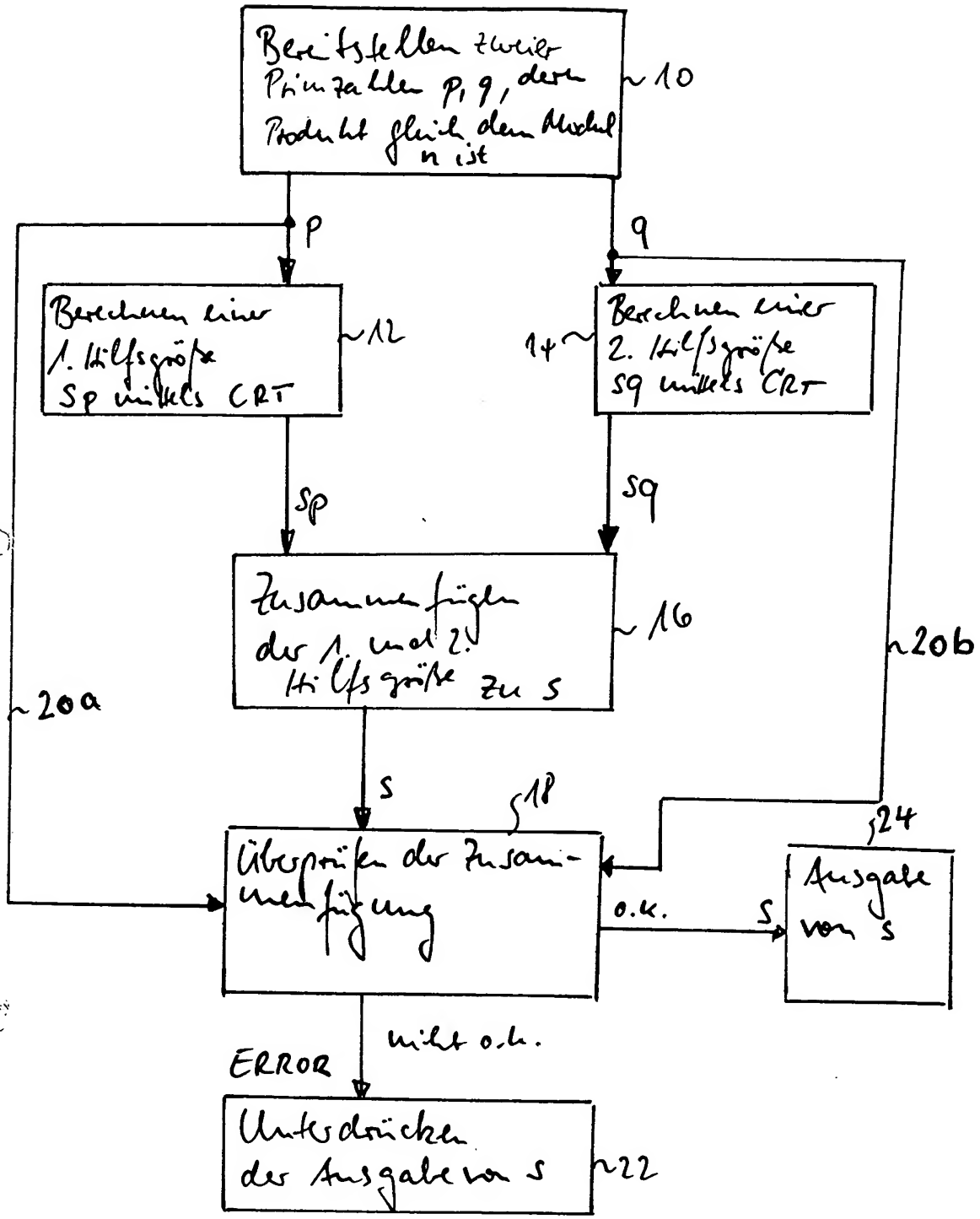

Fig. 1

Figur zur Zusammenfassung

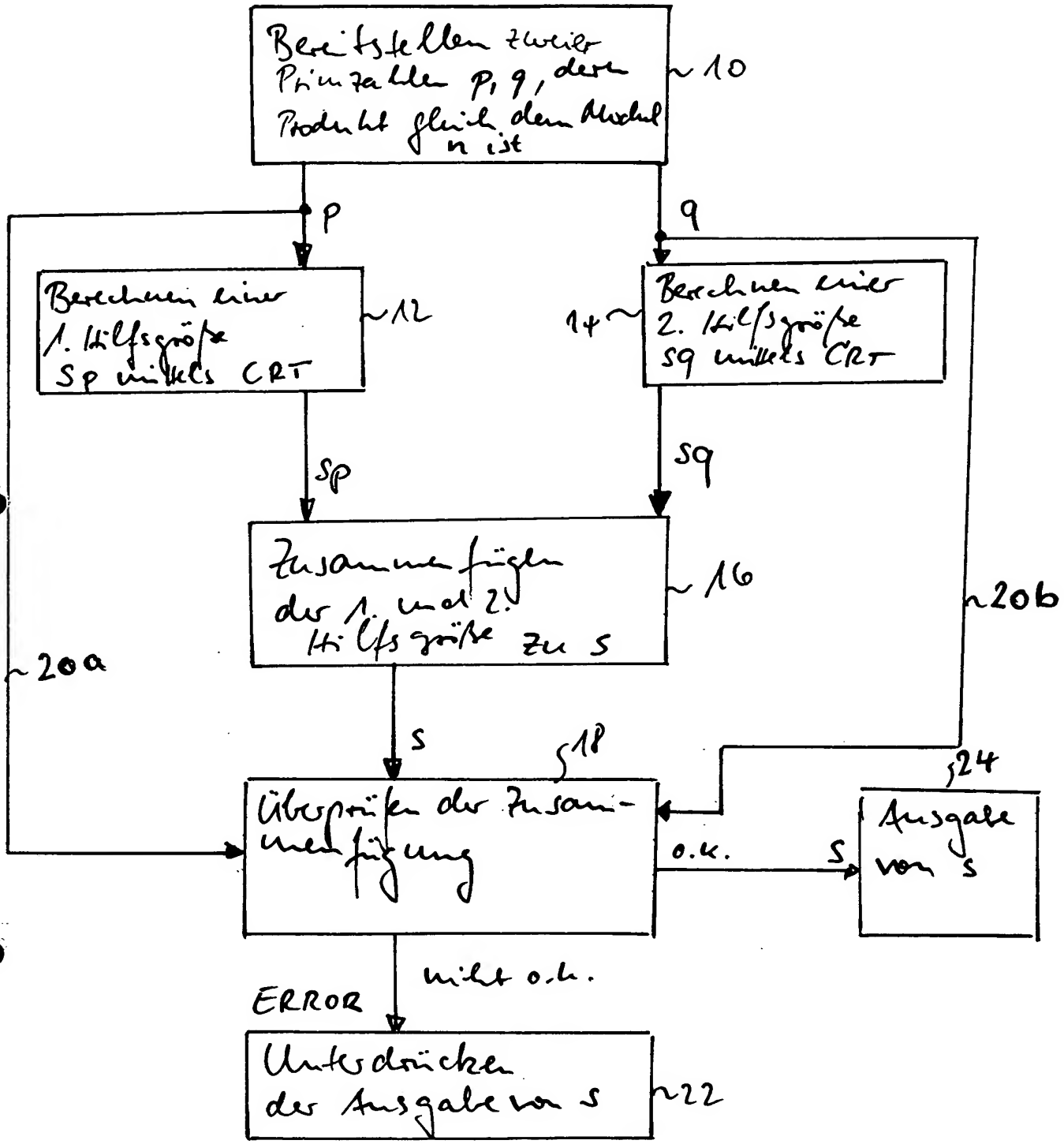
Bereitstellen zweier Primzahlen p, q, deren Produkt gleich dem Modul n ist
10
p
q
Berechnen einer 1. Hilfsgröße sp mittels CRT
12
Berechnen einer 2. Hilfsgröße sq mittels CRT
14
sp
sq
Zusammenfügen der 1. und 2. Hilfsgröße zu s
16
20a
20b
s
18
24
Überprüfen der Zusammenfügung
o.k.
s
Ausgabe von s
nicht o.k.
ERROR
Unterdrücken der Ausgabe von s
22

Fig. 1

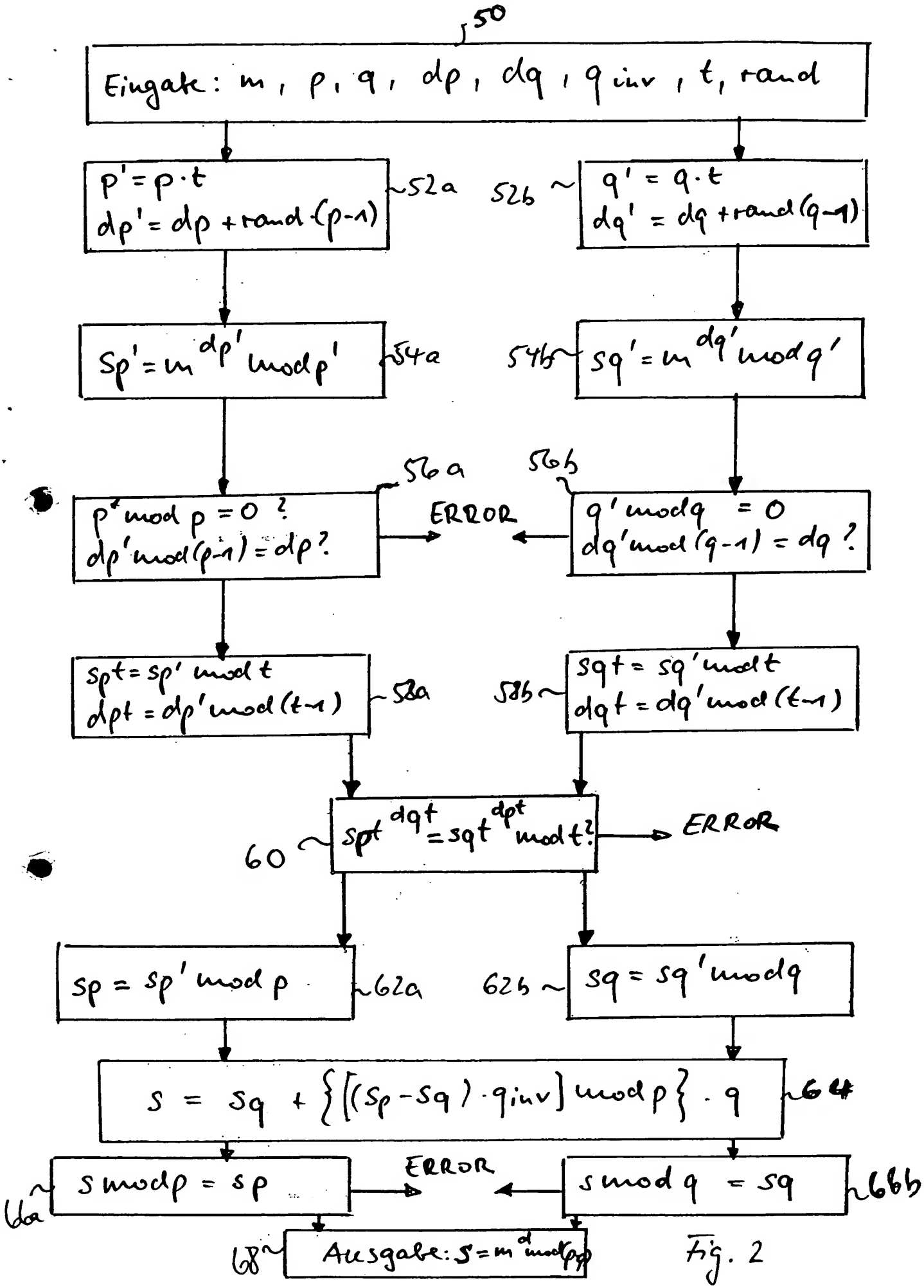
50
Eingabe: m, p, q, dp, dq, qinv, t, rand
p' = p · t
dp' = dp + rand · (p-1)
52a
52b
q' = q · t
dq' = dq + rand (q-1)
sp' = m^dp' mod p'
54a
54b
sq' = m^dq' mod q'
56a
56b
p' mod p = 0 ?
dp' mod (p-1) = dp ?
ERROR
q' mod q = 0
dq' mod (q-1) = dq ?
spt = sp' mod t
dpt = dp' mod (t-1)
58a
58b
sqt = sq' mod t
dqt = dq' mod (t-1)
60
spt^dqt = sqt^dpt mod t ?
ERROR
sp = sp' mod p
62a
62b
sq = sq' mod q
s = sq + {[(sp - sq) · qinv] mod p} · q
64
s mod p = sp
66a
ERROR
s mod q = sq
66b
68
Ausgabe: S = m^d mod pq

Fig. 2

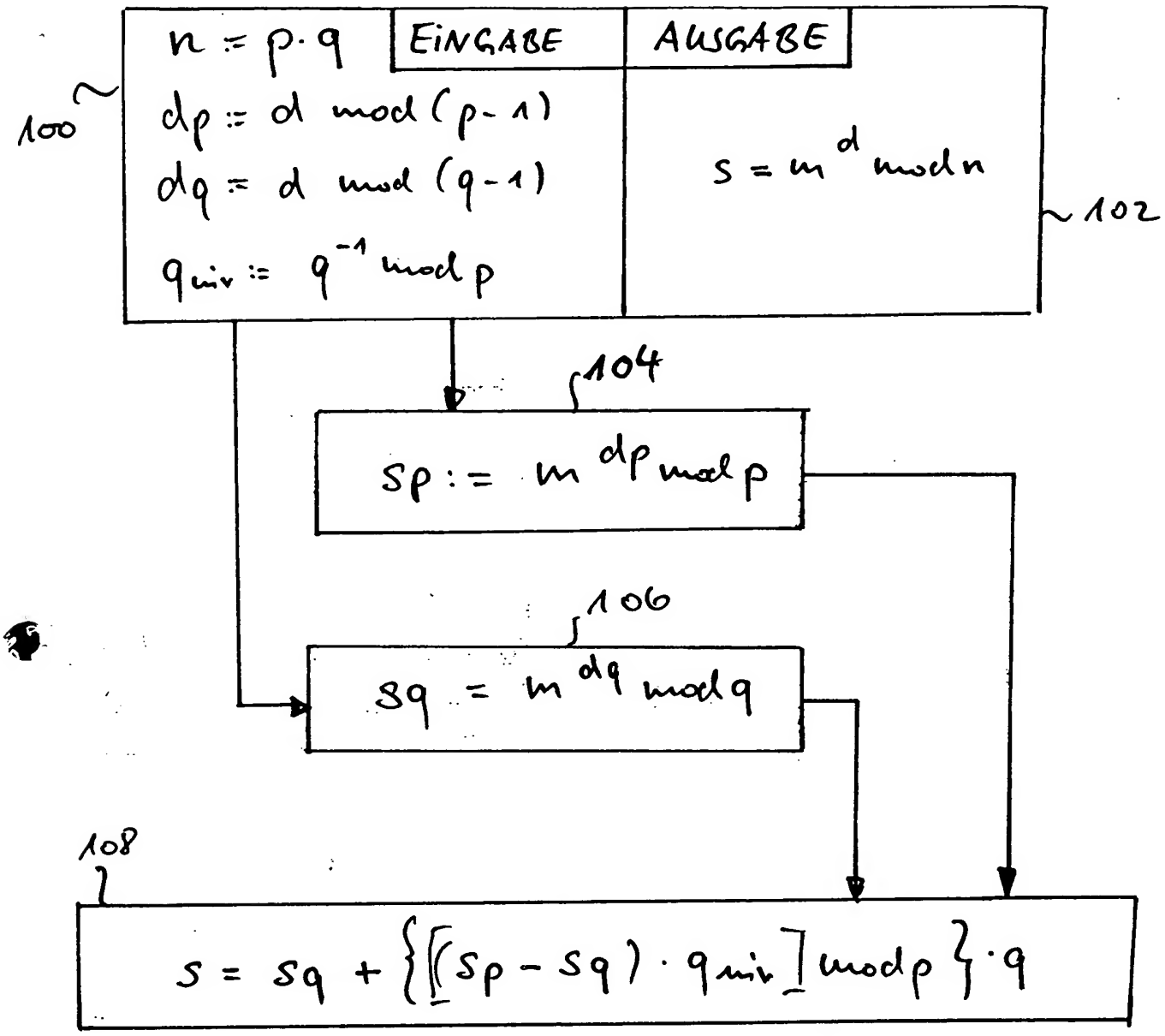

Fig. 3 (Stand der Technik)